LU3IN017

Document mi-parcours

Maxime Chen

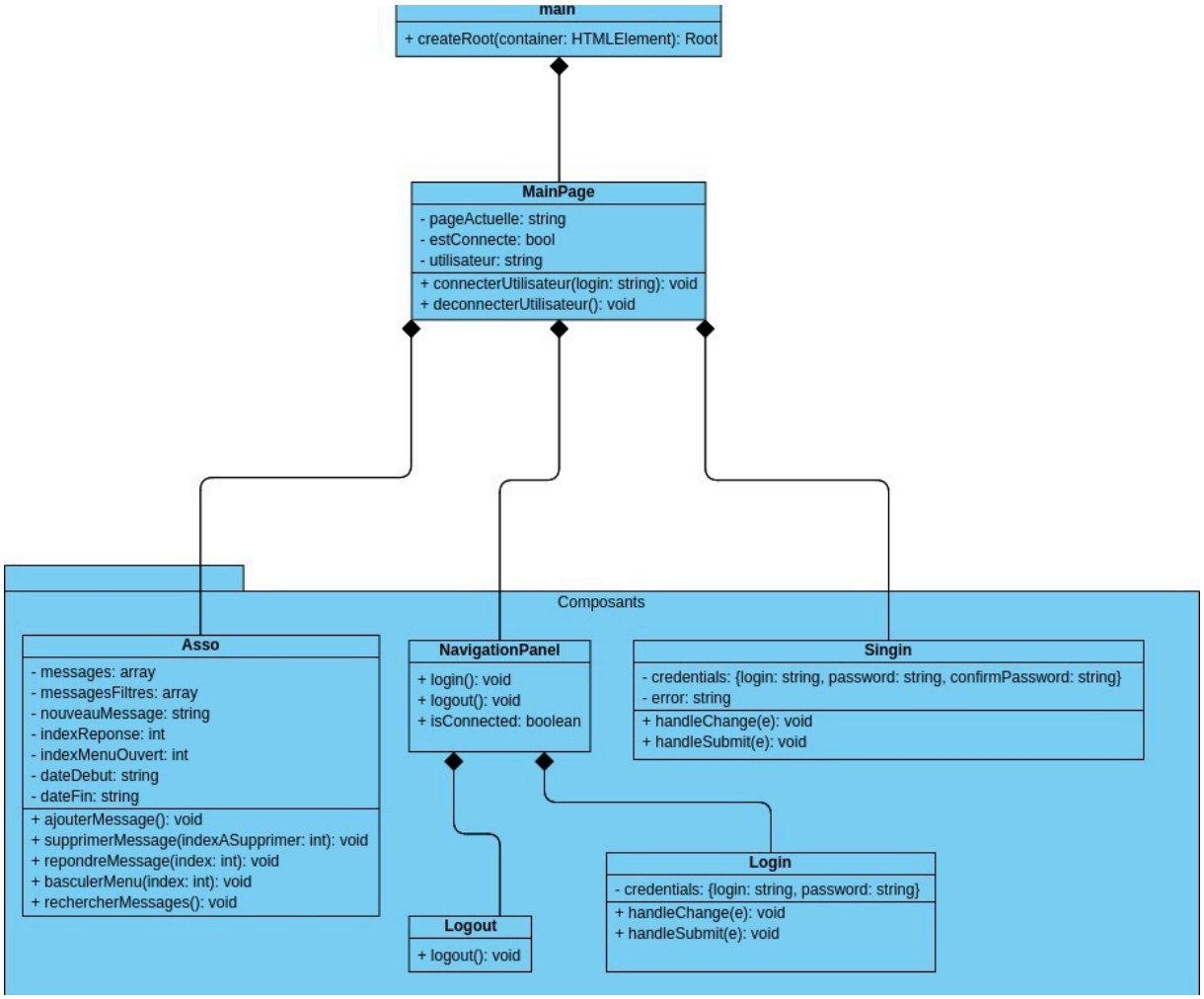
ALi EL Boukili

14 Mars, 2025

Introduction

1. Graphe des dépendances mutuelles entre composants
2. Liste des composants avec quelques explications et ce dont ils ont besoin pour fonctionner

1. Graphe des dépendances



2. Liste des Composants

1. MainPage.jsx

Rôle: Ce composant gère l'affichage de différentes pages d'une application, en permettant à l'utilisateur de se connecter, de se déconnecter, et d'afficher une page d'association (Asso) une fois connecté.

Props : (Aucun)

États:

- **pageActuelle** (chaîne) : Définit la page actuellement affichée. Par défaut, elle est "login", et elle peut changer pour "page_asso" après une connexion réussie.
- **estConnecte** (booléen) : Indique si l'utilisateur est connecté ou non.
- **utilisateur** (chaîne) : Contient le nom de l'utilisateur actuellement connecté.

Liste des composants internes:

- **Asso** : Affiché lorsque l'utilisateur est connecté et que la page actuelle est "page_asso".
- **NavigationPanel** : Affiche un panneau de navigation avec des options de connexion et déconnexion.
- **Signin** : Formulaire de connexion pour entrer les informations d'identification de l'utilisateur.

Pour fonctionner correctement, il a besoin des composants Asso, NavigationPanel, et Signin importés et définis dans le projet.

---

2. NavigationPanel.jsx

Rôle: Ce composant gère l'affichage de l'interface de navigation en fonction de l'état de connexion de l'utilisateur, en affichant soit un bouton de connexion (composant **Login**) soit un bouton de déconnexion (composant **Logout**), selon que l'utilisateur est connecté ou non.

Props:

- **login** : une fonction qui sera appelée pour initier la connexion (présentée à l'utilisateur si il n'est pas connecté).
- **logout** : une fonction qui sera appelée pour effectuer la déconnexion (présentée à l'utilisateur si il est connecté).
- **isConnected** : un booléen qui indique si l'utilisateur est connecté ou non. Si true, le composant affichera **Logout**, sinon il affichera **Login**.

États:

Le composant ne gère pas d'état interne spécifique. Il dépend uniquement des props qui lui sont passées (pas de gestion d'état via **useState** ou autre). Il affiche une interface en fonction de l'état de l'utilisateur (**isConnected**).

Composants inclus:

- **Login** : composant pour gérer la connexion de l'utilisateur.
- **Logout** : composant pour gérer la déconnexion de l'utilisateur.

---

3. Login.jsx

Rôle: Ce composant permet d'afficher un formulaire de connexion où l'utilisateur peut entrer son login et son mot de passe, puis soumettre ces informations à une fonction de connexion passée en prop.

Props:

- **login** : C'est une fonction passée en prop qui sera appelée lors de la soumission du formulaire, avec le login comme argument. Si cette prop n'est pas fournie, le composant ne fonctionnera pas correctement, car il n'aura pas de logique pour gérer la connexion.

États:

- **credentials** : Un objet d'état qui contient deux propriétés `login` et `password` pour stocker les valeurs saisies par l'utilisateur dans les champs de formulaire. Il est initialisé avec des chaînes vides (`""`) pour les deux valeurs.

Composants inclus (Aucun)

---

4. Logout.jsx

Rôle: Ce composant permet d'afficher un bouton qui, lorsqu'il est cliqué, déclenche la déconnexion de l'utilisateur en appelant la fonction **logout** passée en prop.

Props:

- **logout** : Une fonction qui est exécutée lorsque l'utilisateur clique sur le bouton de déconnexion. Elle doit être fournie par le composant parent.

États: Ce composant n'a pas d'état propre, car il se contente de rendre un bouton qui appelle une fonction donnée sans gérer de données internes.

Composants inclus (Aucun)

---

5. Signin.jsx

Rôle : Ce composant **Signin** permet de réaliser un formulaire d'inscription avec vérification que les mots de passes correspondent, et affiche un message d'erreur en cas de non-correspondance.

Props (Aucune)

États:

- **credentials** : Un objet contenant les valeurs du formulaire, à savoir le **login**, **password** et **confirmPassword**. Par défaut, chaque champ est une chaîne vide (**{ login: "", password: "", confirmPassword: "" }**).
- **error** : Une chaîne qui contient un message d'erreur si les mots de passe ne correspondent pas. Par défaut, cette valeur est une chaîne vide (**""**).

Composants inclus (Aucun)

6. Asso.jsx

Rôle :

Ce composant permet de gérer et d'afficher une liste de messages, avec des fonctionnalités de recherche par date, d'ajout de messages, de réponse à un message et de suppression d'un message.

Fonctionnalité : Il permet de réaliser les actions suivantes

- Ajouter un nouveau message
- Répondre à un message
- Supprimer un message
- Filtrer les messages selon une plage de dates
- Ouvrir un menu contextuel pour interagir avec chaque message (répondre ou supprimer)
- Afficher une liste de messages filtrée par la recherche de dates

Props:

- **logout :** Fonction à appeler pour se déconnecter de l'application. Il n'a pas de valeur par défaut.
- **user** : Représente l'utilisateur actuel, utilisé pour afficher l'auteur des messages. Il n'a pas de valeur par défaut.

États:

messages : Liste de tous les messages, initialisée comme un tableau vide.

messagesFiltres : Liste des messages filtrés selon la date de début et de fin spécifiées, initialisée avec messages.

nouveauMessage : Contenu du message en cours d'écriture, initialisé à une chaîne vide.

indexReponse : Indice du message auquel on souhaite répondre, initialisé à null.

indexMenuOuvert : Indice du message pour lequel le menu contextuel est ouvert, initialisé à null.

dateDebut : Date de début pour la recherche des messages, initialisée à une chaîne vide.

dateFin : Date de fin pour la recherche des messages, initialisée à une chaîne vide.

Composants inclus: (Aucun)